Taste of Fully Homomorphic Encryption

Preflight:

- What is FHE?
- LWE & RLWE
- Gadget Decomposition
- Another FHE scheme: Ring-GSW
- External Product

What is FHE?

Homomorphic encryption allows some computation (addition, scalar multiplication, ct-ct multiplication) directly on ciphertexts without first having to decrypt it.

Partially Homomorphic Encryption support only one of those possible operation. RSA is an example:

$$\mathrm{Enc}(m_1) \cdot \mathrm{Enc}(m_2) = m_1^e \cdot m_2^e = (m_1 \cdot m_2)^e = \mathrm{Enc}(m_1 \cdot m_2) \tag{1}$$

FHE supports Addition AND Scalar Multiplicaiton:

$$\begin{cases} \mathrm{Enc}(m_1) + \mathrm{Enc}(m_2) = \mathrm{Enc}(m_1 + m_2) \\ \mathrm{Enc}(m) \cdot c = \mathrm{Enc}(m \cdot c) \end{cases} \tag{2}$$

Fancy! And it exsists!

LWE & Ring-LWE

> *Hiding secrets by adding some noise.*

Learning With Error (LWE)

Given a random vector $a \in \mathbb{Z}_q^n$, a secret key $s \in \mathbb{B}^n, \mathbb{B} = \{0, 1\}$, a small plaintext message m, and a small noise e:

$$\text{LWE}(m) = (a, a \cdot s + m + e) \tag{3}$$

n decides the security level we want. The decryption is also straight forward in high-level:

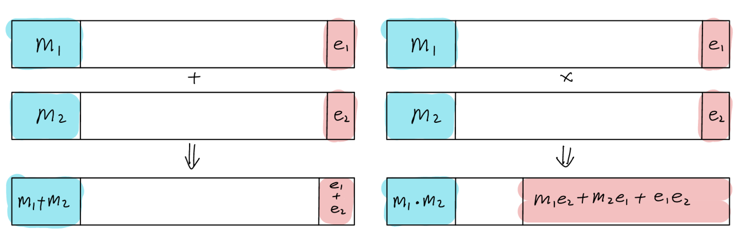
$$\text{Dec}(\ (a, b)\ ) = \lfloor b - a \cdot s \rceil \tag{4}$$

Think of this: if we always encrypt $m = 0$. A "learning without error" scheme can be easily be solved by setting up n linear equations. However, adding this error makes it very hard to solve. (Oded Regev?)

> *The definition I give here is not completely accurate. To encrypt m, one should sometimes scale it up or "shift to the left" so that the lower bits are reserved for noise.*
>
> ***Toy example**: say $q = 2^{32}$, then we can use a 32 bit unsigned integer for each number. Suppose we allow m to have 4 bits, and the rest $32 - 4 = 28$ lower bits are reserved for our noise.*

Additions and scalar mulitiplications are intuitive. ct-ct multiplications needs special design. It's possible!

BFV is a RLWE scheme that supports ct-ct multiplication.
$(a_1, b_1) \cdot (a_2, b_2) = (a_1 b_1, a_1 b_2 + a_2 b_1, b_1 b_2)$.



Ring Learning With Error (RLWE)

Ring variant of LWE. Instead of having a as a vector, we upgrade all vector addition and scalar multiplication are upgraded to polynomial multiplications and additions. Now, we have
$a \in \mathbb{Z}_q[x]/(x^n+1), m \in \mathbb{Z}_t[x]/(x^n+1), s \in \mathbb{B}[x]/(x^n+1)$. n is the polynomial degree. q and t are coefficient modulus for ciphertext and plaintext.

In LWE, we increase the security level by increasing the vector size. Here we increase the polynomial degree. Overall, RLWE is more efficient:

- Each polynomial is huge. So, we can put more information into a single polynomial. Example: $\log q = 124, \log t = 60, n = 4096$.
- " LWE problems tend to require rather large key sizes, typically on the order of n^2." (Regev's survey) To use LWE, typically need n linear equations with errors, each of them has size n key. In RLWE, you only need n coefficients.
- Fast Fourier Transform and Number Theoretic Transform can be applied to polynomials. It makes computation faster!

The problem with noise growth

Addition has additive noise growth, multiplication has multiplicative noise growth. This is bad because we cannot perform this computation many times...

Good news: there is a way to make multiplications have additive noise growth.

Gadget Decomposition

> *How do you calculate 473×128 by hand?*

Simple gadget decomposition (special case):

For a message $m \in \mathbb{Z}$, we encrypt it by scaling it to different powers:

$$\text{Enc}(m) = m \cdot (10^{\ell-1}, \ldots, 10^0) \tag{5}$$

This creates a vector of size ℓ for some chosen number ℓ. Now, if we want to multiply this encrypted message with a constant C, we can calculate the inner product between the decomposed C and this "encrypted" value. Just like what we have learnt in primary school. I.e., we can decompose C to $\text{Decomp}(C) = (C_{\ell-1}, \ldots, C_0)$ such that

$$C = \sum_{i=0}^{\ell-1} C_i 10^i \tag{6}$$

Then the multiplication becomes:

$$C \cdot m = \langle \text{Decomp}(C), \text{Enc}(m) \rangle \tag{7}$$

Toy Example

$m = 6, C = 3405, k = 4$

$$\begin{aligned} \text{Decomp}(C) &= (3, 4, 0, 5) \\ \text{Enc}(m) &= (6000, 600, 60, 6) \\ C \cdot m &= 3 \cdot 6000 + 4 \cdot 600 + 5 \cdot 6 = 20430 \end{aligned}$$

Generalization

Instead of 10, we can use larger base ($B = 256$ for example). Then the gadget looks like $\vec{g} = (B^{\ell-1}, \ldots, B^0)$. Another level of generalization looks like:

$$G = I_k \otimes \vec{g} = \left(\begin{array}{c|c|c} B^{\ell-1} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ B^0 & \cdots & 0 \\ \hline \vdots & \ddots & \vdots \\ \hline 0 & \cdots & B^{\ell-1} \\ \vdots & \ddots & \vdots \\ 0 & \cdots & B^0 \end{array}\right)$$

It turns out that this big G here is also a gadget by definition 3.1 in Building an Efficient Lattice Gadget Toolkit: Subgaussian Sampling and More if we treat each row as an element.

Definition 3.1 *For any finite additive group A, an A-gadget of size w and quality β is a vector $\mathbf{g} \in A^w$ such that any group element $u \in A$ can be written as an integer combination $u = \sum_i g_i \cdot x_i$ where $\mathbf{x} = (x_1, \ldots, x_w)$ has norm at most $\|\mathbf{x}\| \leq \beta$.*

There is a very good property of gadget decomposition:

$$\|\mathrm{Decomp}_G(v) \cdot G - v\|_\infty \leq \epsilon \tag{8}$$

It is also good at controlling the noise:

$$\langle \mathrm{Decomp}(C), \mathrm{Enc}(m) \rangle = \sum_{i=0}^{\ell-1} C_i \cdot \mathrm{Enc}(m)_i \tag{9}$$

If each row has some error e, a direct multiplication has $O(C \cdot e)$ noise growth, while this special multiplication has $O(\log(C) \cdot e)$.

Ring-GSW

This is another FHE scheme. The special thing about this scheme is that it uses RLWE and a special gadget matrix during the encryption stage.

$$\mathrm{RGSW}(m) = Z + m \cdot G \qquad Z = (\underbrace{\mathrm{RLWE}(0), \ldots, \mathrm{RLWE}(0)}_{2\ell}), G = I_2 \otimes \vec{g}$$

This is a Ring-GSW sample that encrypts the message m (without scaling by any factor). This is a special case of RGSW (otherwise $G = I_k \otimes \vec{g}$ for any desired k. Check TFHE). And, we also have 2ℓ rows of RLWE(0). This encryption is straight forward. However, I don't know any decryption methods. Why? Note that some gadget values are very small compared to the ciphertext coefficient modulus, which means $m \cdot g_i$ can be as small as the noise...

External Product

> *Additive noise growth for ct-ct multiplication!!!*

External product using only one gadget decomposition:

$$\mathrm{RGSW} \boxdot \mathrm{RLWE} \to \mathrm{RLWE}$$
$$(A, b) \mapsto A \boxdot b = \mathrm{Decomp}_G(b) \cdot A$$

English: we first decompose the RLWE ciphertext, and then multiply it with RGSW.

Proof sketch

Let $\mathrm{msg}(A) = \mu_A, \mathrm{msg}(b) = \mu_b$. By definition of RLWE,
$b = (a, a \cdot s + \mu_b + e) = (a, a \cdot s + 0 + e) + (0, \mu_b) = z_b + (0, \mu_b)$.

Let $\boldsymbol{u} = \mathrm{Decomp}_G(b)$ below.

$$\begin{aligned} A \boxdot b = \boldsymbol{u} \cdot A &= \boldsymbol{u} \cdot (Z_A + \mu_A \cdot G) \\ &= \boldsymbol{u} \cdot Z_A + \mu_A \cdot (\boldsymbol{u} \cdot G) \\ &= \boldsymbol{u} \cdot Z_A + \mu_A \cdot (\epsilon + b) \\ &= \boldsymbol{u} \cdot Z_A + \mu_A \cdot \epsilon + \mu_A \cdot (z_b + (0, \mu_b)) \\ &= \boldsymbol{u} \cdot Z_A + \mu_A \cdot \epsilon + \mu_A \cdot z_b + (0, \mu_A \cdot \mu_b) \end{aligned}$$

Recall, decryption is to calculate the linear equation $\mathrm{Dec}(\ (a, b)\ ) = b - a \cdot s$.
Then, taking the expectation, everything goes to zero except $\mu_A \cdot \mu_b$.

Why this is good?

Check the noise growth! Roughly this way:

$$\|\mathrm{Err}(A \boxdot b)\|_\infty \leq \|\boldsymbol{u} \cdot \mathrm{Err}(A)\|_\infty + |\mu_A| \cdot \epsilon + |\mu_A| \cdot \mathrm{Err}(b)$$
$$\text{Roughly: } O(B \cdot \mathrm{Err}(A) + |\mu_A| \cdot \mathrm{Err}(b))$$

If we have small message μ_A, then this multiplication is roughly free! Ok, but why? I must quote this sentence I learnt from Jeremy Kun: "This is useful when the noise growth is asymmetric in the two arguments, and so basically you make the noise-heavy part as small as possible and move the powers of 2 to the other side."

Then the essence is that we separate RLWE, which is very sensitive to scaling, to smaller parts, and then perform this "bit-by-bit" multiplication. We can do this because we have carefully designed this RGSW scheme so that it stores enough information for all powers of B, saving this scaling for RLWE.

References:

I found Jeremy Kun recently. He had some amazing blogs on FHE:

- A High-Level Technical Overview of Fully Homomorphic Encryption
- The Gadget Decomposition in FHE

TFHE: Faster Fully Homomorphic Encryption: Bootstrapping in less than 0.1 Seconds

Wiki pages:

- RSA
- Homomorphic Encryption

First LWE: On lattices, learning with errors, random linear codes, and cryptography

Gadget: Building an Efficient Lattice Gadget Toolkit: Subgaussian Sampling and More

O. Regev, "The Learning with Errors Problem (Invited Survey)," 2010 IEEE 25th Annual Conference on Computational Complexity, Cambridge, MA, USA, 2010, pp. 191-204, doi: 10.1109/CCC.2010.26.
keywords: {Equations;Cryptography;Lattices;Zinc;Computer errors;Polynomials;Computational complexity;Decoding;Computer science;Application software;learning with errors;lattice-based cryptography},